PROGRAMACIÓN III

Practica Final

PRESENTACIÓN

Nombre

Russbell Smith Montero Turbi

Matricula

2022-0548

Maestro

Kelyn Tejada Belliard

INDICE

PORTADA DE TRABAJO

NOMBRE DEL PROYECTO

School Project.

TECNOLOGIAS PARA APLICAR

Mediante Frontend. Mediante la aplicación ReactJS, la aplicación HTML y la aplicación CSS. Entorno: Node js y Express. Anexo de Datos: Estimación de datos en MongoDB. Servidor Web Web. "Nginx", alias "Nginx". Net core, C#,Diseño de contenido: Mediante firewall, Sistemas de Pruebas Automatizadas Se requiere una selección de datos. A: a/GitHub.com Formato: Github Tasks, Selenium…..

OBJETIVO DEL PROYECTO

Diseño de un sistema educativo integral que posibilite la gestión eficiente de los usuarios, materias, calificaciones y asistencia, brindando una plataforma accesible y segura para los docentes, docentes y estudiantes.

ALCANCE DEL PROYECTO

Sistema de Gestión de Usuarios (Creación, Modificación y eliminación) , administración de Materias y Cursos. Registro de Asesoría, Administración y Visualización de Calificaciones. Seguridad y autenticación.

CRONOGRAMA DEL PROYECTO:

Semana 1-2: Análisis de Requerimientos y Diseño del Sistema

Semana 3-4: Desarrollo del Módulo de Autenticación y Gestión de Usuarios

Semana 5-6: Desarrollo del Módulo de Materias y Calificaciones

Semana 7-8: Desarrollo del Módulo de Asistencia

Semana 9-10: Pruebas y Validación

Semana 11-12: Implementación y Despliegue

DEFINIR CLARAMENTE LO QUE EL SISTEMA EN UN PRIMER RELEASE VA A PODER HACER.

En el primer release, el sistema será capaz de: Autenticar usuarios. Permitir la creación, modificación y eliminación de usuarios. Registrar y gestionar materias. Registrar asistencia. Gestionar calificaciones.

DEFINIR REQUERIMIENTOS DEL SISTEMA PARA EL PRIMER RELEASE

Frecuentemente: Se procede a la autenticación de los usuarios. Administración de usuarios (administradores, profesores, estudiantes) La gestión de temáticas. Registro de asesoría. Dirección de evaluaciones.

No se dispone de ninguna función. Seguridad en la autenticación y el control de datos. Rendimiento excepcional para asistir a múltiples usuarios simultáneos. La interfaz usuario es intuitiva y accesible.

DEFINIR TAREAS A EJECUTAR

Análisis de Requerimientos: Identificar y documentar necesidades del sistema.

Diseño de la Arquitectura: Crear diagramas de arquitectura del sistema.

Desarrollo del Frontend: Implementar la interfaz de usuario.

Desarrollo del Backend: Implementar la lógica del servidor y la integración con la base de datos.

Pruebas Unitarias: Desarrollar pruebas unitarias para asegurar la funcionalidad.

Pruebas de Integración: Validar la integración entre módulos del sistema.

Documentación: Redactar la documentación técnica y de usuario.

DEFINIR EL EQUIPO DE TRABAJO (ROLES, HABILIDADES, ETC)

Product Owner: Responsable de la visión del producto y de priorizar el backlog.

Scrum Master: Facilita la metodología Scrum y elimina impedimentos.

Frontend Developer: Desarrolla la interfaz de usuario.

Backend Developer: Desarrolla la lógica del servidor y la base de datos.

QA Engineer: Automatiza y ejecuta pruebas de calidad.

DevOps Engineer: Configura la infraestructura y gestiona el CI/CD.

Stakeholders: Directivos y profesores del colegio modelo.

HERRAMIENTAS QUE USARÍAN

Gestión de Proyectos: Jira o Trello

Control de Versiones: Git y GitHub

CI/CD: GitHub Actions

Pruebas: Selenium con NUnit

Documentación: Confluence o Google Docs

Comunicación: Slack o Microsoft Teams

DEFINIR LAS ÉPICAS

Épica 1: Gestión de Usuarios

Épica 2: Gestión de Materias

Épica 3: Registro de Asistencia

Épica 4: Gestión de Calificaciones

Épica 5: Autenticación y Seguridad

DEFINIR LAS FECHAS PARA CADA CEREMONIA DE SCRUM.

Sprint Planning: Todos los jueves 1 de agosto.

Daily Standup: Todos los días a las viernes.

Sprint Review: Cada dos sábados..

Sprint Retrospective: Domingos y lunes.

AFINAR AL MENOS 10 HISTORIAS DE USUARIOS (CRITERIOS DE ACEPTACIÓN OBLIGATORIOS).

H1. Como administrador del sistema, quiero poder crear nuevos usuarios (profesores, estudiantes) para gestionar sus accesos al sistema.

CA1. El sistema debe permitir agregar usuarios, ya sea para profesores o estudiantes.

H2. Como profesor, quiero registrar la asistencia de los estudiantes para llevar un control diario de sus asistencias.

CA2: El sistema debe permitir al usuario registrar la asistencia de los estudiantes.

H3. Como estudiante, quiero ver mis calificaciones en línea para estar al tanto de mi rendimiento académico.

CA3. El sistema debe permitir a los usuarios estudiantes visualizar su calificación.

H4. Como administrador, quiero poder actualizar la información de los profesores para mantener los datos actualizados.

CA4. El sistema debe permitir actualizar la información de los profesores

H5. Como estudiante, quiero poder cambiar mi contraseña para asegurar la privacidad de mi cuenta. Dicha funcionalidad se trabajará en otro sprint.

CA5. El sistema debe mostrar una opción para cambiar contraseña y esta no debe redireccionar a ningún sitio.

H6. Como administrador, quiero poder eliminar usuarios obsoletos del sistema para mantener la base de datos actualizada.

CA6. El sistema debe permitir a los administradores eliminar usuarios.

H7. Como profesor, quiero poder agregar nuevas materias para gestionar las clases que imparto.

CA7. El sistema debe permitir agregar nuevas materias.

H8. Como estudiante, quiero poder inscribirme en nuevas materias para poder cursarlas en el siguiente semestre.

CA8. El sistema debe permitir agregar nuevas materias.

H9. Como administrador, quiero poder generar reportes de asistencia para evaluar la participación de los estudiantes.

CA9. El sistema debe permitir registrar la asistencia de los estudiantes.

H10. Como profesor, quiero poder actualizar las calificaciones de mis estudiantes para reflejar su desempeño académico.

CA10.El sistema debe permitir actualizar las calificaciones de los estudiantes.

REALIZAR UN PLAN DE PRUEBAS QUE CONTENGA LO SIGUIENTE PUNTOS, CON AL MENOS 20 CASOS DE PRUEBAS QUE SEAN OBLIGATORIOS DENTRO DE CUALQUIER TIPO DE SISTEMA (EJEMPLO INICIO DE SESIÓN DE USUARIO, CREACIÓN DE USUARIOS, CREACIÓN DE ROLES, ETC.):

1 Creación exitosa de un nuevo usuario.

Dado: Que el administrador accede al módulo de usuarios.

Cuando: Ingrese todos los datos requeridos y presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de confirmación y el nuevo usuario debería estar disponible en la lista de usuarios.

2 Error en la creación de usuario por falta de datos obligatorios

Dado: Que el administrador intenta crear un usuario sin completar todos los campos obligatorios.

Cuando: Presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de error indicando los campos faltantes.

3 Registro exitoso de asistencia.

Dado: Que el profesor accede al módulo de asistencia.

Cuando: Seleccione una clase y marque la asistencia de los estudiantes.

Entonces: El sistema debería registrar la asistencia y mostrar un mensaje de confirmación.

4 Error al registrar asistencia por clase no seleccionada.

Dado: Que el profesor intenta registrar la asistencia sin seleccionar una clase

Cuando: Presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de error indicando que debe seleccionar una clase.

5 Visualización exitosa de calificaciones.

Dado: Que el estudiante accede a su perfil.

Cuando: Seleccione la opción de "Ver Calificaciones".

Entonces: El sistema debería mostrar las calificaciones correspondientes a cada materia.

6 Error al visualizar calificaciones debido a la falta de datos.

Dado: Que el estudiante intenta acceder a la sección de calificaciones.

Cuando: No hay calificaciones disponibles.

Entonces: El sistema debería mostrar un mensaje indicando que no hay calificaciones disponibles.

7 Actualización exitosa de la información del profesor.

Dado: Que el administrador accede al módulo de profesores.

Cuando: Modifique la información de un profesor y presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de confirmación y la información actualizada debería reflejarse.

8 Error al actualizar información por datos no válidos.

Dado: Que el administrador intenta actualizar la información con datos incorrectos (e.g., formato de correo no válido).

Cuando: Presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de error indicando los campos incorrectos.

9 Cambio exitoso de contraseña.

Dado: Que el estudiante accede a su perfil.

Cuando: Ingrese la contraseña actual, la nueva contraseña y confirme la nueva contraseña.

Entonces: El sistema debería permitir el cambio y mostrar un mensaje de confirmación.

10 Error en el cambio de contraseña por contraseñas no coincidentes.

Dado: Que el estudiante ingresa contraseñas nuevas que no coinciden.

Cuando: Presione "Guardar".

Entonces: El sistema debería mostrar un mensaje de error indicando que las contraseñas no coinciden.

LISTA DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

1. Requerimientos Funcionales

Registro de usuarios (HU1):

El sistema debe permitir a los administradores crear nuevos usuarios, ya sean profesores o estudiantes.

El sistema debe validar que los datos ingresados para la creación de usuarios sean correctos y completos.

Registro de asistencia (HU2):

El sistema debe permitir a los profesores registrar la asistencia de los estudiantes diariamente.

El sistema debe guardar y actualizar la asistencia registrada en la base de datos.

Visualización de calificaciones (HU3):

El sistema debe permitir a los estudiantes visualizar sus calificaciones en línea.

Las calificaciones deben ser accesibles solo para el estudiante correspondiente.

Actualización de información de profesores (HU4):

El sistema debe permitir a los administradores actualizar la información personal y profesional de los profesores.

Los cambios realizados deben reflejarse inmediatamente en la base de datos.

Prototipo de cambio de contraseña (HU5):

El sistema debe mostrar una opción para que los estudiantes puedan iniciar el proceso de cambio de contraseña.

Aunque no se implementará completamente en este release, el sistema debe indicar que la funcionalidad se trabajará en un sprint futuro.

Eliminación de usuarios (HU6):

El sistema debe permitir a los administradores eliminar usuarios obsoletos o inactivos.

La eliminación debe garantizar que todos los datos relacionados con el usuario sean removidos de forma segura.

Gestión de materias (HU7):

El sistema debe permitir a los profesores agregar nuevas materias que impartirán.

Las materias agregadas deben ser visibles para los estudiantes durante el proceso de inscripción.

Inscripción en materias (HU8):

El sistema debe permitir a los estudiantes inscribirse en nuevas materias para el próximo semestre.

La inscripción debe actualizarse en tiempo real en la base de datos.

Generación de reportes de asistencia (HU9):

El sistema debe permitir a los administradores generar reportes de asistencia de los estudiantes.

Los reportes deben ser exportables en formatos comunes, como PDF o Excel.

Actualización de calificaciones (HU10):

El sistema debe permitir a los profesores actualizar las calificaciones de sus estudiantes.

Las actualizaciones deben ser visibles para los estudiantes una vez publicadas.

2. Requerimientos No Funcionales

Seguridad:

El sistema debe garantizar la autenticación y autorización segura para proteger los datos de los usuarios.

Debe utilizarse cifrado para el almacenamiento de contraseñas y datos sensibles.

Rendimiento:

El sistema debe ser capaz de manejar múltiples usuarios concurrentes sin una degradación perceptible en el rendimiento.

Las operaciones críticas deben completarse en menos de 2 segundos.

Usabilidad:

La interfaz debe ser intuitiva, fácil de usar y accesible para todos los usuarios, independientemente de su nivel técnico.

Debe ofrecerse documentación o tutoriales para facilitar la adaptación de los usuarios al sistema.

Escalabilidad:

El sistema debe estar diseñado para escalar horizontalmente para soportar un número creciente de usuarios y datos.

Debe ser posible añadir recursos adicionales (servidores, bases de datos) sin necesidad de rediseñar el sistema.

Compatibilidad:

El sistema debe ser compatible con los principales navegadores web y dispositivos móviles.

Debe asegurar una experiencia de usuario consistente en diferentes plataformas.

Mantenibilidad:

El código del sistema debe estar bien documentado para facilitar su mantenimiento y futuras actualizaciones.

Deben utilizarse prácticas de desarrollo que favorezcan la modularidad y el uso de pruebas automatizadas.

Disponibilidad:

El sistema debe estar disponible el 99.9% del tiempo durante el horario escolar.

Debe existir un plan de contingencia para minimizar el tiempo de inactividad en caso de fallas.

Auditoría y Log:

El sistema debe registrar todas las acciones críticas (creación, eliminación, actualización) para permitir auditorías.

Los logs deben ser almacenados de manera segura y accesibles solo para administradores autorizados.

Backup y Recuperación:

El sistema debe implementar un proceso regular de backups de la base de datos.

Debe ser posible recuperar la información en caso de un fallo o pérdida de datos.

Localización e Internacionalización:

El sistema debe soportar múltiples idiomas y formatos de fecha/hora.

La interfaz debe adaptarse a diferentes configuraciones regionales sin necesidad de cambios de código.

DEFINIR CUALES SON LOS CRITERIOS DE ACEPTACIÓN DE LAS PRUEBAS.

Las funcionalidades deben operar según los criterios de aceptación definidos en las historias de usuario.

La interfaz debe ser intuitiva y fácil de usar.

DEFINIR CUALES SON LOS CRITERIOS DE RECHAZO DE LAS PRUEBAS.

La prueba será rechazada si se detectan errores críticos en las funcionalidades o si el sistema no cumple con los requerimientos de rendimiento.

COMENTAR SOBRE LAS HERRAMIENTAS DE PRUEBAS QUE ESTARÍAN USANDO Y JUSTIFICAR RESPUESTA.

Selenium con NUnit: Para pruebas automatizadas de la interfaz de usuario.

Jira: Para seguimiento de incidencias.

Justificación: Selenium permite automatizar pruebas en aplicaciones web, y NUnit facilita la organización y ejecución de pruebas en un entorno C#.

ESTIMACIÓN DEL TIEMPO PARA LA EJECUCIÓN DE PRUEBAS

Preparación del entorno: 2 días

Escritura de scripts de pruebas: 5 días

Ejecución de pruebas: 3 días

Revisión y ajuste: 2 días

ELABORAR UNA PLANTILLA CON LOS EQUIPOS DE PRUEBAS Y SUS RESPONSABILIDADES.

QA Engineer 1: Pruebas de autenticación.

QA Engineer 2: Pruebas de gestión de usuarios.

QA Engineer 3: Pruebas de integración y rendimiento.

ELABORAR UN PLAN DE AUTOMATIZACIÓN DE PRUEBAS.

Fase 1: Automatizar pruebas de autenticación y gestión de usuarios.

Fase 2: Automatizar pruebas de registro de asistencia y gestión de materias.

Fase 3: Ejecución de pruebas de integración y rendimiento.

CONCLUSIONES

El desarrollo de un sistema escolar utilizando la metodología Agile-Scrum y la definición de requerimientos tanto funcionales como no funcionales permite una planificación detallada y estructurada del proyecto. Al afinar las historias de usuario y definir criterios de aceptación claros, se asegura que el producto final cumpla con las expectativas y necesidades de los usuarios. Además, al abordar aspectos no funcionales como seguridad, rendimiento, y escalabilidad, se garantiza que el sistema no solo sea funcional, sino también robusto y sostenible a largo plazo.

La utilización de Agile-Scrum facilita la adaptación a cambios y mejora continua a lo largo del desarrollo, lo que es crucial en entornos educativos donde las necesidades pueden evolucionar rápidamente. Por otro lado, la importancia de un plan de pruebas bien estructurado no puede ser subestimada, ya que asegura la calidad del sistema y minimiza errores que podrían impactar negativamente la experiencia del usuario.

BIBLIOGRAFIAS

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. Scrum.org.

Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley Professional.

Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Software Development, 9(8), 28-35.

Ambler, S. W. (2004). Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. Wiley.

Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional.

Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing (3rd ed.). Wiley.

IEEE. (1998). IEEE Standard for Software Test Documentation. IEEE Std 829-1998.

Beck, K. (2003). Test Driven Development: By Example. Addison-Wesley Professional.